ECSE 2920: Design Methodology

Deliverable 3

Group 11

01/27/2025

Part 1: Switch-Controlled LED

Requirements: Program a switch controlling an LED. Describe Debounce method and draw the circuit.

In order to program a switch to control an LED we need to do the following...

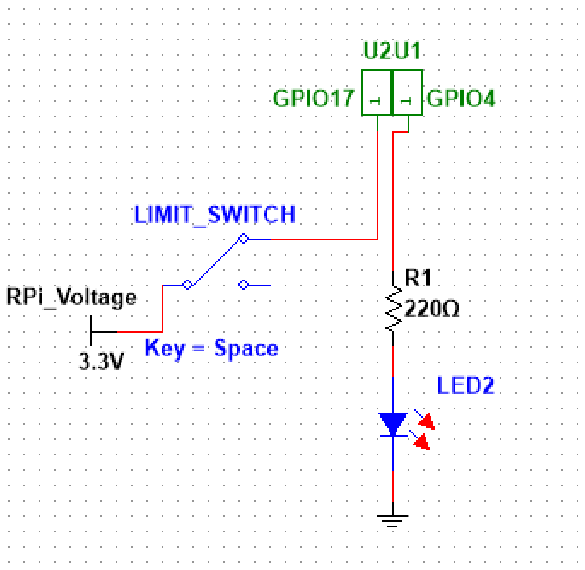
- Lookup the pin setup of our limit switch
- Understand how to construct LED and Button objects in our python library
- Understand debouncing and decide how to implement it.

Debounce Method can be done through hardware or software:

1. Software Debounce (Preferred):
   - This method uses a delay or timing mechanism to ignore spurious transitions during the bounce period
2. Hardware Debounce:
   - This involves using a capacitor and resistor circuit to filter out the bouncing signals (ex, an RC low-pass filter)

We decided to utilize software Debounce instead of hardware.

Picture(s):



(Image Showing Constructed circuit of a switch controlling and LED)

Key Design Decision(s)

- What did your team clarify about the design?
  - Power/Ground MUST come from terminals, not pins
- What were the competing choices in the design?
  - Given the design constraints, there wasn't much design variation

    - We did have to decide between hardware or software debouncing
- Ultimately what did your team choose and why?
    - We decided to use our limit switch to test the limit switch’s hardware too.
    - We also decided to do software debouncing because it was programmable as a constructor parameter in our python library.

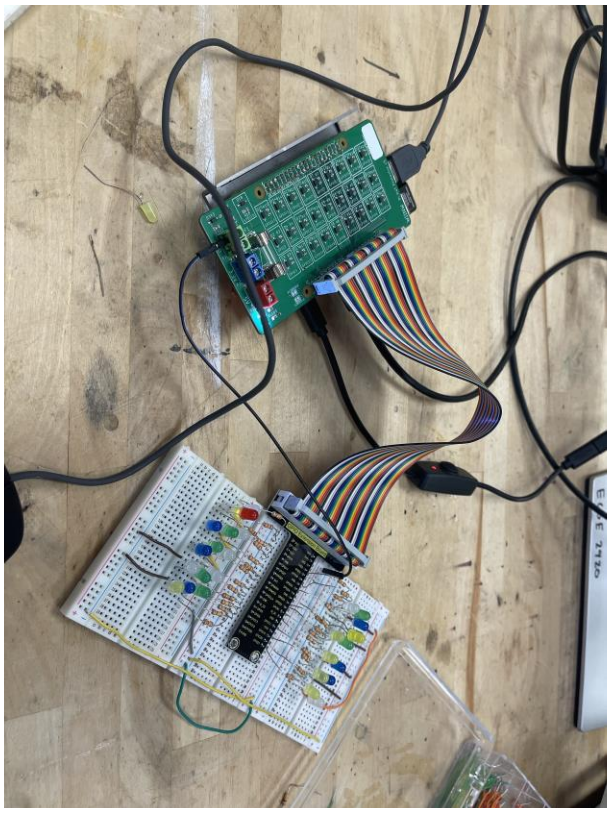
TEST… Test… test

- What aspects of the design need to be tested?
    - Software?
        - Does our chosen python library work with GPIO inputs and outputs?
        - Does the software debouncing work?
    - Hardware?
        - Does our limit switch work? How does it work?
- Who is responsible for testing? Michael & Dawson
    - Tests ran?
        - We ran multiple tests with different debounce amounts and found 0.25 seconds to be a responsive yet safe amount for software debouncing.
    - Conclusions from testing?
        - Everything worked as planned in a responsive manner.

We were able to confirm that our python library works, the GPIO pins work in an I/O context, and we were able to better understand how to use software debouncing when it needs to be used in the real project.

<u>Part 2: GPIO Tester</u>
Requirements: Write a start-on-launch program to check all GPIOs using the GPIO Tester

Because we have already tested how to assign a pin to an LED object in our python program, this section had a lot less uncharted territory involved. The most difficult part of this check was the time it took to wire everything. Once everything was done and I set the SPI pins off, every GPIO Pin was able to give a correct output signal except for GPIO 23.



Key Design Decision(s)

- What did your team clarify about the design?
    - Power needed to come from the terminals
    - Each GPIO needed to be identified as working
- What were the competing choices in the design?
    We had two ideas for the main GPIO test.
    - Each GPIO gets an LED
    - All GPIOs feed into LEDs with a larger delay
- Ultimately what did your team choose and why?

    - While it took more time and resources, we decided to give each GPIO an LED and a 220 Ohm Resistor. This would help us clearly define the status of each GPIO pin.

TEST… Test… test

- What aspects of the design need to be tested?
    - Software?
        - We needed to test if our python library really worked on both inputs and outputs
    - Hardware?
        - We needed to check if each GPIO correctly worked
- Who is responsible for testing? Michael
    - Tests ran? We ran tests that ran the GPIOs in multiple different orders and then arranged the order to prepare for the Deliverable 3 GPIO Test.
    - Conclusions from testing?   GPIO 23 is broken, but everything else works as planned.

After talking to trudgen, we were told to mark the pi with tape in order to notify testers that GPIO 23 was nonfunctional. We believe that our hardware/ software combination not only works but is efficient. Because we had to iterate through each GPIO pin, we had to approach the programming side with a "work smart not hard" mentality. This allowed us to get acclimated to loops and arrays in python and use those tools to minimize the amount of code.

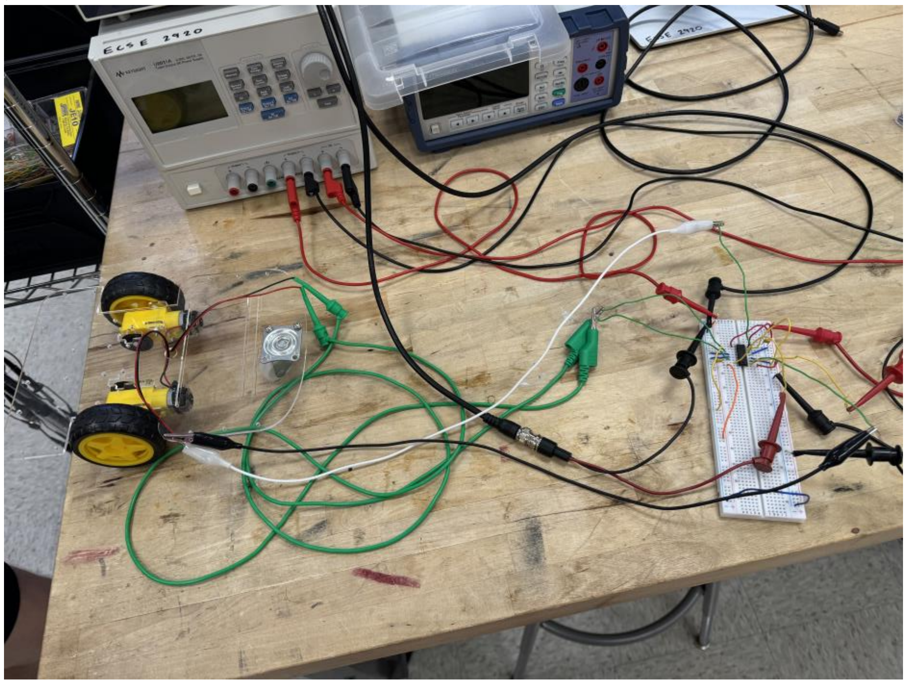
<u>Part 3: DC Motor Operation</u>

Requirements: Wiggle the two DC motors (forward, backward, and one by one)

What determines the speed of the motor?

The speed of the motor is determined by the duty cycle. This is because the higher the duty cycle the higher the average voltage is delivered to the motor.

Picture(s)



Key Design Decision(s)

- We did not change anything about the design of the circuit diagram in deliverable 2 to build the circuit and test the H-Bridge and motors.

TEST… Test… test

- What aspects of the design need to be tested?
    - We had to test the logic of our circuit to see if the motor would actually spin forward and backward when we expected it to.
    - We also had to test the duty cycle to see if it affected the speed of the motor
- Who is responsible for testing?
    - Tests ran?
        - While the motor was spinning, we adjusted the duty cycle on the wavegen. To see if the motor speed changed. The duty cycle started at 50% and I turned it down to 30%. When I turned it down to 30% the motor slowed down. I then turned the duty cycle up to 70% and then the motor sped up to a faster speed than that of the 50%.

    - We also tested the logic of the motors spinning. We believed that when the 1A (input of H-Bridge) was high and 2A (input of H-Bride) was low that the motor would spin forward, and vice versa for backwards. To test this, we connected 2A to ground and 1A to a 3.3 Vpp 200Hz square wave. When we did this the motor spun forward. When we connected 1A to ground and 2A to the 3.3 Vpp the motor spun backward.
  - Conclusions from testing?
    - The conclusion is that adjusting the duty cycle to a lower percentage will decrease the speed of the motor. Adjusting the duty cycle to a higher percentage will increase the speed of the motor.
    - The conclusion is that our logic was correct. When 1A was high and 2A low the motor will spin forward. When 2A is high and 1A is low the motor will spin backward.

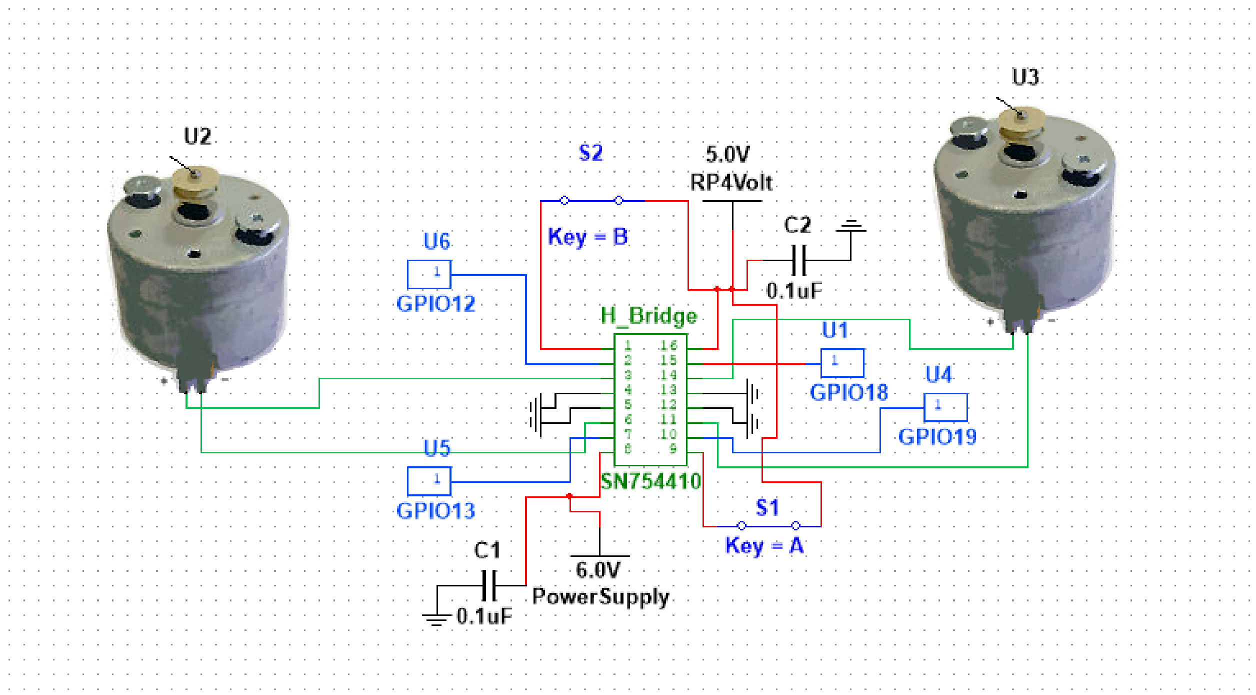
summary/part conclusion (make sure you address all parts of the requirements)

In summary the duty cycle affects the speed of the motor by increasing the average voltage. The higher the duty cycle the higher the RPM. The logic of the motor worked as expected. When one input was high and the other low the motor would spin would way and vice versa for the other.

<u>Part 4: Schematic Design</u>

Requirements: Create a schematic design that lays out the RP4, H-bridge with DC motors, and switches

Picture(s)



For dc motors, the left wire is positive, and the right wire is negative.

Key Design Decision(s)

- What did your team clarify about the design?
    - The team clarified that switches would be between enable pin and 5V from the raspberry pi.
    - The team clarified how the second motor would be wired.
    - The team clarified that some GPIO pins had been changed around because the team no longer used GPIOs to control enable.
    - The team also clarified that we use GPIOs 12, 13, 18, 19 because they all have built in PWM
    - Wires are colored respectively for readability
- What were the competing choices in the design?
    - In our original design we had the enable pin wired to a GPIO so we could control when the motors were enabled. In our new design we have the enable wired to 5V with a dip switch.
    - There were no competing choices for how to wire the second motor.
    - 2 competing decisions were use pins with built in PWM or don't
- Ultimately what did your team choose and why?
    - Our team decided to go with the dip switch and 5V for the enable instead of the GPIO pin, because this way we save space for GPIO pins in the future and having a dip switch allows us to control whether or not the motors are enabled anyway.
    - The team ended up wiring the 2nd motor the same way we wired the first, using 2 GPIO pins connected to the inputs 3A and 4A of the H-Bridge.

- GPIOs 12, 13, 18, and 19 are wired to pins 2,7, 10, and 15 on the H-Bridge respectively because all these GPIOs default low this way our motors don’t unexpectedly run without code running.
- The team decided to go with pins that have built in PWM because if we did not we would have to use software to control the PWM

summary/part conclusion (make sure you address all parts of the requirements)

In summary the only thing that the team changed about the design was what we did with the enable. In our new design we wired the enables for both sides of the H-Bridge to 5V through a dip switch. We did this to save space for GPIOs in the future, and the dip switch allows us to control whether or not the motors are enabled. The team also changed around some of the GPIOs because we no longer needed them to control the enable and used GPIOs with built in PWM (pulse width modulation).

Part 5: Microphone Characterization

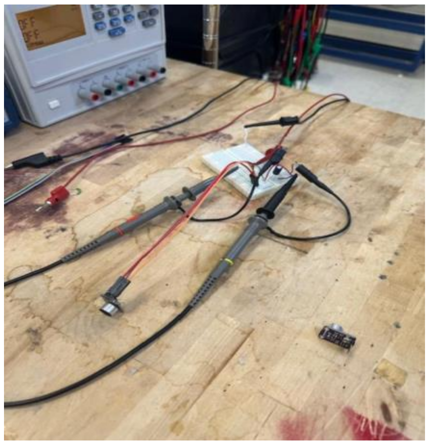
Requirements: Characterize the microphone. See how sensitive it is and make sure it is characterized enough for it to do what it needs to do with the Audio Car.

To characterize the microphone, we set up a simple circuit where we had a 3.3V DC source connected to the input of the microphone, ground connected to the microphone's ground, and a 100 uF capacitor connected to the output of the microphone. We connected the capacitor because we have a DC source instead of AC, so we needed AC coupling.

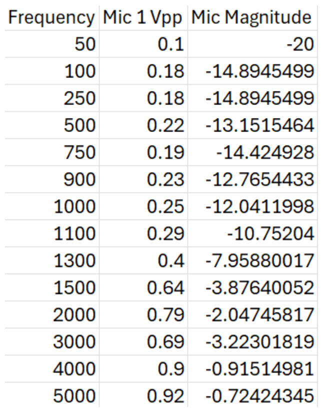
To receive measurements for the microphone we connected 2 probes from an oscilloscope to the input and output of our circuit (output being after the capacitor). We then played different audio frequencies and recorded the |Vout/Vin| response, we recorded these values in an excel document so we could create a bode plot once all measurements were received.

While doing this we also ensured the gain on both of our microphones were as close as possible, we did this by changing the gain on the second microphone until our data we were receiving was the same as the first microphone's

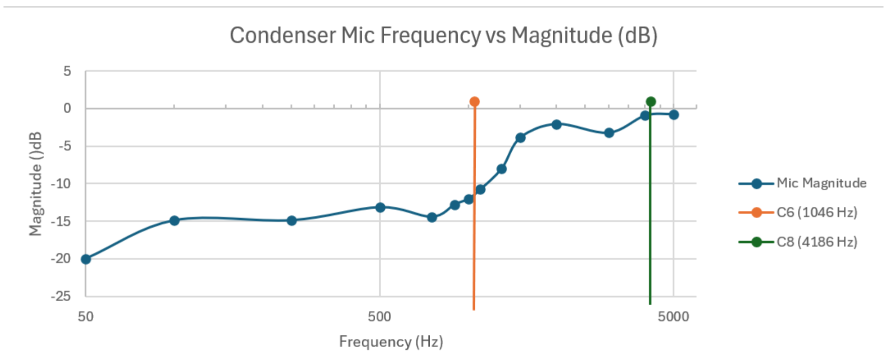
Picture(s)



(Circuit setup for microphone characterization)



Frequency	Mic 1 Vpp	Mic Magnitude
50	0.1	-20
100	0.18	-14.8945499
250	0.18	-14.8945499
500	0.22	-13.1515464
750	0.19	-14.424928
900	0.23	-12.7654433
1000	0.25	-12.0411998
1100	0.29	-10.75204
1300	0.4	-7.95880017
1500	0.64	-3.87640052
2000	0.79	-2.04745817
3000	0.69	-3.22301819
4000	0.9	-0.91514981
5000	0.92	-0.72424345



(Our Data, and Bode plot showing our results from our testing)

Key Design Decision(s)

- What did your team clarify about the design?
    - We all understood that a big part of this project is filters, we just needed to decide on which filters to include.
- What were the competing choices in the design?
    - After we received all the data there weren't any competing choices in the design, we all agreed on one filter.
- Ultimately what did your team choose and why?

    - Ultimately, we concluded that we're going to need a band-pass filter. This is a suitable choice because it passes frequencies within a specific range while attenuating frequencies outside this range, and it helps isolate the target frequencies (C6 and C*)

TEST… Test… test

- What aspects of the design need to be tested?
    - Software?
        - The software testing for our filters will need to be conducted later on, not currently.
    - Hardware?
        - We plan to create our filter and test the response of the microphone when connected to it, this will be done in the upcoming weeks.

summary/part conclusion (make sure you address all parts of the requirements)
We successfully characterized the microphone by constructing a simple circuit powered by a 3.3V DC source, with an AC coupling capacitor to remove the DC bias. Using an oscilloscope, we measured the |Vout/Vin| response for various frequencies and created a Bode plot to analyze its sensitivity. The team concluded that a band-pass filter is the best solution to isolate the target frequencies (C6 and C8) while attenuating unwanted signals. Further testing of the filter and software integration will be conducted to ensure the microphone meets the requirements for the audio car.

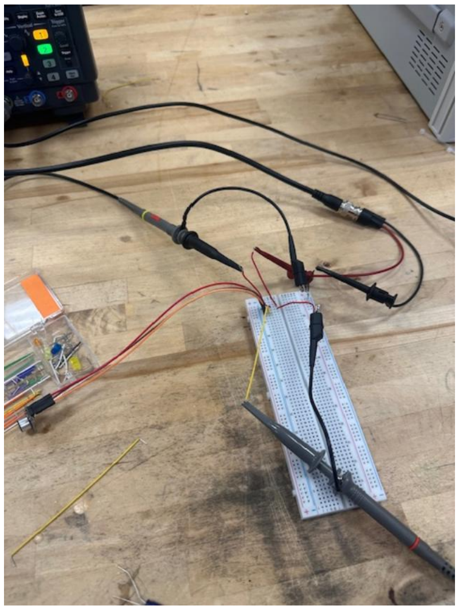
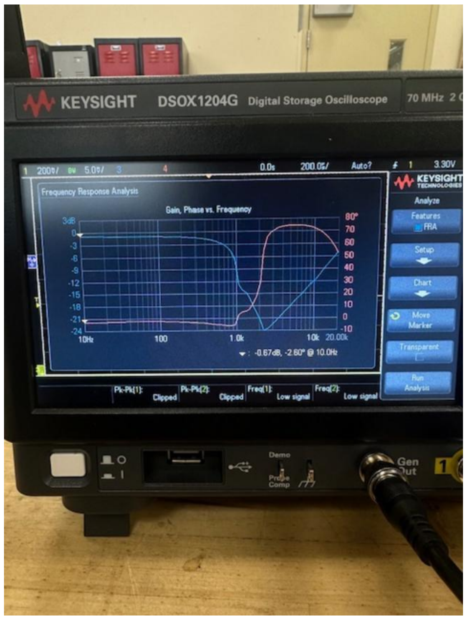
<u>Part 6: Spectrum Analyzer</u>

Requirements: Practice using a spectrum analyzer

To prepare to use the spectrum analyzer we constructed a circuit around the microphone. Unlike when characterizing it, the input of the microphone was from Gen Out on the oscilloscope, this is an AC signal so the capacitor at the output also is not needed. We ensured that the microphone was grounded, and then we connected two probes, one to input, and one to output of the microphone, grounding each probe.

We then ran the spectrum analyzer from 10Hz to 20,000Hz and after messing with the scaling setting for the graph, received a graph that represents the results best.

Picture(s)



KEYSIGHT
DSOX1204G Digital Storage Oscilloscope
Frequency Response Analysis
Gain, Phase vs. Frequency
Analyze
Features
Setup
Chart
Run Analysis
Gen Out

(Left: Circuit setup for conducting frequency response analysis; Right: The results of our frequency response analysis)

Key Design Decision(s)

- These results directly align with our results from the microphone characterization. Therefore, giving us the same design decisions, which basically tells us that we're going to need a band-pass filter to distinguish our target frequencies

The same testing will need to be done as with the microphone characterization deliverable, which is connecting the microphone to the band-pass filter and reviewing the results. This will be conducted in the upcoming weeks.

summary/part conclusion (make sure you address all parts of the requirements)
We successfully practiced using the spectrum analyzer by constructing a circuit around the microphone, using the oscilloscope's Gen Our as the input signal and eliminating the need for the output capacitor. By running the spectrum analyzer from 10Hz to 20,000 Hz and adjusting the scaling, we generated a graph that aligned with the results from the microphone characterization. These consistent findings confirmed the need for a band-pass filter to isolate the target frequencies. Further testing will involve connecting the microphone to the band-pass filter and verifying its performance in upcoming weeks.

Conclusion and Participation (REQUIRED)

1. **Include a selfie/photo from your group meeting/zoom call.**

2. **When did you meet? 1/17 and 1/21**
3. **Who was present?**

   Due to MLK weekend, availability was off, so we met in groups of 2
   - Dawson and Michael: 1/17
   - Fin and Cade: 1/21
4. **Who was not present?**
   - Everyone was present in at least one meeting this week
5. **What were the main ideas discussed or major decisions (1-2 sentences/bullet points)**
   - Future filter design for the microphone. After reviewing the data we received we talked over the different filters that we could use, and settled on what we think will be the best design.
   - The overall layout of our car and its components. Since all the components need to have a place on the car itself, we took some time to think over where everything should be laid out before getting farther into the design process.